第8章 数据库编程

沈明玉

合肥工业大学 Hefei University of Technology 计算机与信息学院

第8章 数据库编程

本章内容：

8.1 嵌入式SQL

8.2 过程化SQL

8.3 存储过程与函数

8.4 SQL中的事务处理

第8章 数据库编程

8.1 嵌入式SQL

- SQL语言提供了交互式SQL和嵌入式SQL两种使用方法。
- 当一个程序既要访问数据库，又要处理数据时，将SQL语言嵌入到程序设计语言（宿主语言）中，将SQL语言访问数据库的功能和宿主语言的数据处理功能相结合，这种使用SQL的方法称为**嵌入式SQL**。
- 嵌入式SQL的使用需要解决的问题：
  - ① 宿主语言的编译器不能识别SQL，如何将嵌有SQL语句的宿主语言编译成可执行代码？
  - ② 宿主语言与DBMS之间如何传递数据和信息？
  - ③ 如何解决元组集合数据对宿主语言变量的赋值？

嵌入式SQL（续）

8.1.1 嵌入式SQL的处理过程

- 第 ① 个问题的解决方法

  预编译（如图所示）

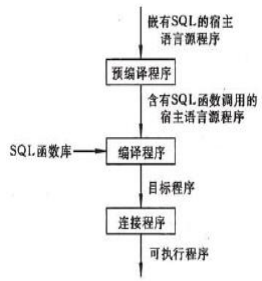
- 在宿主语言中标注SQL语句的方法

C语言程序：

EXEC SQL <SQL语句>;

Java语言程序：

SQL {<SQL语句>};



8.1 嵌入式SQL

8.1.2 嵌入式SQL语句与主语言之间的通信

- 解决第②③两个问题
- 解决问题的三种手段：
  - **SQL通信区（SQLCA）**：向主语言传递SQL语句的执行状态
  - **主变量（共享变量）**：向SQL语句提供参数，接受SQL语句的查询结果
  - **游标**：解决SQL查询结果的多对一赋值（向主变量）问题

8.1 嵌入式SQL

1. SQL通信区

✓ SQLCA的用途
- SQLCA是一个具有特点数据结构的缓冲区；
- 描述系统当前的工作状态和运行环境。

✓ SQLCA使用方法
- 定义SQLCA：Exec SQL Include SQLCA
- 使用SQLCA：变量SQLCODE，为0-执行成功，否则表示出错。
- 属于SQL的内容，主语言可以访问。

8.1 嵌入式SQL

2.主变量（共享变量）

- ✓ 主变量是主语言的变量，SQL与主语言均可对其进行赋值（共享）；
- ✓ 宿主语言与SQL语言之间通过共享变量进行数据传送（双向）；
- ✓ 每个主变量均可附带一个指示变量，用来说明主变量的值是否为空或是否被截断等；
- ✓ 共享变量和指示变量在SQL语句的说明部分（Declare Section）进行说明，说明后在SQL语句中可引用（引用时需加':'号，以区分数据库自身的变量，如：属性）。

8.1 嵌入式SQL

3. 游标

- ◘ **游标cursor**
  - ✓ 游标是为了解决元组集合对主变量的赋值问题（多对一）；
  - ✓ 游标是为用户开设的一个数据缓冲区；
  - ✓ 采用类似指针的技术定位某一个元组。
- ◘ **游标的使用方法**
  - ✓ 定义游标：Exec SQL declare <游标名> cursor for <查询子句>;
  - ✓ 打开游标：Exec SQL open <游标名>;
  - ✓ 推进游标：Exec SQL fetch <游标名> into <主变量表>;
  - ✓ 关闭游标：Exec SQL close <游标名>;

8.1 嵌入式SQL

4. 建立和关闭数据库连接

- ◘ **建立数据库连接**

Exec SQL Connect TO target [AS connection-name] [User user-name];

- ✓ *target*是要连接的数据库服务器；
- ✓常见的服务器标识串，如<dbname>@<hostname>:<port>
- ✓ *connect-name*是可选的连接名，也可使用DEFAULT

- ◘ **关闭数据库连接**

Exec SQL DISConnect [connection|CURRENT];

- ✓ Connection是已建的链接

8.1 嵌入式SQL

5. 程序实例

[例]依次检查某个系的学生记录，交互式更新某些学生年龄。

```
EXEC SQL BEGIN DECLARE SECTION;                    /*主变量说明开始*/
    char deptname[20];
    char HSno[20];
    char HSname[20];
    char HSsex[2];
    int HSage;
    int NEWAGE;
EXEC SQL END DECLARE SECTION;                      /*主变量说明结束*/
EXEC SQL INCLUDE sqlca;                    /*定义SQL通信区*/
```


8.1 嵌入式SQL

```
int main(void)
{   int    count = 0;
    char  yn;
    printf("Please choose the department name(CS/MA/IS): ");
    scanf("%s", deptname);                          /*为主变量deptname赋值*/
    EXEC SQL CONNECT TO TEST@localhost:54321 USER
            "SYSTEM" /"MANAGER";                    /*连接数据库TEST*/
    EXEC SQL DECLARE SX CURSOR FOR                  /*定义游标*/
            SELECT Sno, Sname, Ssex, Sage
            FROM Student
            WHERE SDept = :deptname;
    EXEC SQL OPEN SX;
```


8.1 嵌入式SQL

```
for ( ; ; )
{   EXEC SQL FETCH SX INTO :HSno, :HSname, :HSsex,:HSage;
                                        /*推进游标，将当前数据放入主变量*/
    if (sqlca.sqlcode != 0)
            break;
    if(count++ == 0)
            printf("\n%-10s %-20s %-10s %-10s\n", "Sno", "Sname", "Ssex", "Sage");
    printf("%-10s %-20s %-10s %-10d\n", HSno, HSname, HSsex, HSage);

    printf("UPDATE AGE(y/n)?");          /*询问用户是否要更新该学生的年龄*/
    do {
            scanf("%c",&yn);
    } while(yn != 'N' && yn != 'n' && yn != 'Y' && yn != 'y');
```

```
    if (yn == 'y' || yn == 'Y')
    {
        printf("INPUT NEW AGE:");
        scanf("%d",&NEWAGE);
        EXEC SQL UPDATE Student              /*执行SQL语言的修改语句*/
                 SET Sage = :NEWAGE  WHERE CURRENT OF SX ;
    }                                        /*对当前游标指向的学生年龄进行修改*/
  }
  EXEC SQL CLOSE SX;                         /*关闭游标SX*/
  EXEC SQL COMMIT WORK;                      /*事务提交*/
  EXEC SQL DISCONNECT TEST;                  /*断开数据库连接*/
}
```


8.1.3 不用游标的SQL语句

这类SQL语句包括：说明性语句、数据定义语句、数据控制语句、查询结果为单行的SELECT语句、非CURRENT形式的增删改语句。

```
Exec SQL Begin Declare Section：
    char sno[10], sname[20];
    int sage;
Exec SQL End Declare Section：
```

1. 查询结果为单行的SELECT语句

```
Exec SQL Select sname, sage into :sname, :sage  from  Student
    Where sno=:sno;
```


2. 非CURRENT形式的增删改语句

```
Exec SQL Insert into Students (sno, sname, sage)
            values (:sno, :sname, :sage);
Exec SQL Update Students SET sname=:sname, sage=:sage
            Where sno=:sno;
Exec SQL Delete from SC  Where sno = :sno;
```


8.1.4 使用游标的SQL语句

必须使用游标的SQL语句有：查询结果为多行的SELECT语句、CURRENT形式的UPDATE和DELETE语句。

1. 查询结果为多行的SELECT语句

```
EXEC SQL DECLARE SX CURSOR FOR          /*定义游标*/
    SELECT Sno, Sname, Ssex, Sage
    FROM Student
    WHERE SDept = :deptname;
EXEC SQL OPEN SX;                       /* 打开游标*/
EXEC SQL FETCH SX INTO :HSno, :HSname, :HSsex,:HSage;
                                        /* 推进游标并对主变量赋值*/
EXEC SQL CLOSE SX;                      /*关闭游标SX*/
```


2. CURRENT形式的UPDATE和DELETE语句

WHERE CURRENT of <游标名> ：当前游标指针所指向的元组。

```
EXEC SQL UPDATE Student  SET Sage = :NEWAGE
        WHERE CURRENT OF SX ;
EXEC SQL DELETE FROM  Student   WHERE CURRENT OF SX ;
```

注意：游标中的SELECT语句带有UNION、ORDER或该SELECT语句相当于定义了一个不可更新的视图时，不能使用CURRENT形式的UPDATE和DELETE语句！即：要保证游标中的元组与基本表中元组的对应关系不变！

8.1.5 动态SQL

动态SQL指SQL语言可以在程序中动态构造，形成一个字符串，然后再交给DBMS执行，交给DBMS执行时仍然可以传递变量。

动态SQL支持动态组装SQL语句和动态参数。

1. 使用SQL语句主变量

```
exec sql include sqlca;
exec sql begin declare section;
    char sqltext[]="delete from SC where Cno=\'c210\'";
exec sql end declare section;
.....
exec sql execute  immediate :sqltext;
```

8.1 嵌入式SQL

2. 动态参数

动态参数是SQL语句中的可变元素，使用参数符号（？）表示该位置的数据在运行时设定。

使用动态参数的步骤如下：

① 声明SQL语句主变量：SQL语句主变量的值包含动态参数（？）

② 准备SQL语句（PREPARE）：PREPARE将分析含主变量的SQL语句内容，建立语句中包含的动态参数的内部描述符，并用<语句名>标识它们的整体。

EXEC SQL PREPARE <语句名> FROM < SQL 语句主变量>

③ 执行准备好的语句（EXECUTE）：EXECUTE将SQL语句中分析出的动态参数和主变量或数据常量绑定，作为语句的输入或输出变量。

EXEC SQL EXECUTE <语句名> [INTO <主变量表>] [USING <主变量或常量>] ;

8.1 嵌入式SQL

[例]

```
EXEC SQL begin declare section;
    char sqltext[]="insert into  TEST values(?);";
EXEC SQL end declare section;
……
EXEC SQL PREPARE  mystmt   FROM : sqltext ;          /* 准备语句*/
……
EXEC SQL EXECUTE mystmt  USING  100;                 /* 执行语句*/
EXEC SQL EXECUTE mystmt  USING  200;
```

Hefei University of Technology 计算机与信息学院

第8章 数据库编程

8.2 过程化SQL

✓ **传统的SQL语言是非过程化语言，只能进行数据操作，不能进行数据处理，为了使SQL同时兼顾数据操作和数据处理的能力，DBMS厂商在标准SQL的基础上进行了过程化语言的扩充，如：Oracle的PL/SQL、SQL Server的Transact-SQL、IBM DB2的SQL PL等。**

Hefei University of Technology 计算机与信息学院

8.2 过程化SQL

8.2.1 Transact-SQL

SQL Server用于操作数据库的编程语言为Transaction-SQL，简称**T-SQL**。

T-SQL包括以下4个部分：

- **DDL：**定义和管理数据库及其对象，例如create、alter和drop等。
- **DML：**实现对数据库表各对象的操作，例如insert、update等。
- **DCL：**数据控制语言，实现对数据库进行安全管理和权限管理等控制，例如grant、revoke等。
- **附加的语言元素。**T-SQL的附加语言元素，包括变量、运算符、函数、注释和流程控制语句等。

Hefei University of Technology 计算机与信息学院

8.2 过程化SQL

■ **T-SQL的变量**

(1)局部变量

- 局部变量由用户定义
- 局部变量名称必须以@开始开头
- 局部变量使用declare变量声明语句定义，其语法格式为：

  declare @变量1 [as] datatype, @变量2 [as] datatype …

(2) 全局变量

- 由SQL Server系统定义，通常用来跟踪服务器范围和特定会话期间的信息，不能被用户显式地定义和赋值。
- 全局变量名以@@开头。如: @@error, @@servername, @@rowcount等。

Hefei University of Technology 计算机与信息学院

8.2 过程化SQL

■ **T-SQL的赋值语句**

T-SQL局部变量的赋值有三种方式：

① 在变量定义的时候对其赋值：

declare @变量1 [as] datatype = value,@变量2 [as] datatype = value…

② select赋值语句，其语法格式为：

select @变量1 = 表达式1,@变量2 = 表达式2…

③ set赋值语句，其语法格式为：

set @变量 = 表达式

set赋值语句基本用法和select一样，区别在于一条set赋值语句只能给一个变量赋值，而一条select语句可以给多个变量赋值。

Hefei University of Technology 计算机与信息学院

8.2 过程化SQL

■ **T-SQL的WHILE循环、FOR循环、IF语句、游标**

```
DECLARE @sno  char(10) , @cno  char(6), @grade numeric(3,1)
DECLARE My_Cursor CURSOR  FOR
    ( SELECT * FROM SC  Where  Cno='c209' and Grade<60 )
OPEN My_Cursor;
FETCH NEXT FROM My_Cursor INTO @sno , @cno, @grade ;
WHILE @@FETCH_STATUS = 0
    BEGIN
       IF @grade>=50
          UPDATE SC  SET Grade=60  WHERE CURRENT OF My_Cursor;
       FETCH NEXT FROM My_Cursor INTO @sno , @cno, @grade ;
    END
CLOSE My_Cursor;                          --关闭游标
DEALLOCATE My_Cursor;                     --释放游标
```

8.2 过程化SQL

■ **异常处理**

T-SQL 代码中的错误可使用 TRY...CATCH 构造处理。TRY...CATCH 构造包括两部分：一个 TRY 块和一个 CATCH 块

```
BEGIN TRY
   SELECT 1/0;
END TRY
BEGIN CATCH
   SELECT
   ERROR_NUMBER() AS ErrorNumber,
   ERROR_PROCEDURE() AS ErrorProcedure,
   ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```


8.2.2 PL/SQL

PL/SQL是Oracle对标准SQL的过程化扩充，它将数据库技术和过程化程序设计语言联系起来，可使用循环、分支和嵌套，将SQL的数据操纵功能与过程化语言数据处理功能相结合。

■ **PL/SQL程序的基本结构**

DECLARE—可选部分
　　变量、常量、游标、用户定义异常的声明
BEGIN—必要部分
　　SQL语句和PL/SQL语句构成的执行程序
EXCEPTION—可选部分
　　程序出现异常时，捕捉异常并处理异常
END：—必须部分

■ **PL/SQL的变量**

● **简单变量**

格式：变量名 [constant] 变量类型 [not null] [default 值 | :=值]

```
v_name  varchar2(10);
v_age  constant  number:=20;
v_sex char(2) default '男';
```

● **复合变量**

```
布尔类型：        v_tf  boolean;
type类型：        v_sno student.sno%type;
rowtype类型：     v_sc  sc%rowtype;    (使用：v_sc.sno)
```


■ **PL/SQL的语句**

● **赋值语句**

变量名称 := 表达式

```
v_sno := '20180080';
```

● **条件语句**

IF-THEN，IF-THEN-ELSE和嵌套的IF语句

```
IF ( new.Job = '讲师')  AND  ( new.Sal < 3000 )  THEN
        new.Sal := 3000;
END IF;
```


● **循环语句**

✓ **基本loop循环**

```
Loop
   语句1;
   ……
   exit [when 条件];
End loop;
```

✓ **While loop循环**

```
While 条件
loop
   语句1;
   ……
   语句n;
End loop;
```

✓ **For loop循环**

For 控制变量 in [reverse] 下限..上限

```
loop
   语句1;
   ……
   语句n;
End loop;
```

■ **PL/SQL 的异常处理**

```
EXCEPTION
   WHEN exception_name THEN
      Code for handing exception_name;
   [WHEN another_exception THEN
      Code for handing another_exception];
   [WHEN others THEN
      code for handing any other exception.];
```

Oracle预定义异常：

no_data_found, too_many_rows, storage_error, zero_divide等。

第8章 数据库编程

8.3 存储过程和函数

- 过程化SQL编写的程序块：命名块和匿名块
- 匿名块：不保存，每次运行都要进行编译，其他程序不能调用。
- 命名块：编译后以数据库对象的形式保存在数据库服务器上，可被其他程序调用。
- 存储过程与函数均属于命名块。

8.3 存储过程和函数

8.3.1 存储过程

■ **存储过程：由过程化SQL语句书写的过程，经编译和优化后存储在数据库服务器中，使用时只要调用即可。**

■ **存储过程的优点：**

- ✓ 运行效率高；
- ✓ 降低了客户机和服务器之间的通信量 ；
- ✓ 方便实施企业规则。

8.3 存储过程和函数

1. 创建PL/SQL存储过程

```
Create [or replace] procedure 过程名
   [(argument1 [in|out|in out] type1 , argument2 [in|out|in out] type2, ……)]
{IS | AS}
    <类型、变量的说明>
Begin
    <执行部分>
Exception
    <可选的异常处理部分>
end;
```

注：PL/SQL创建存储过程的例子参见教材存储过程部分！

8.3 存储过程和函数

2. 创建T-SQL存储过程

```
CREATE PROC[EDURE] 存储过程名@参数1 数据类型[=默认值] [INPUT |
    OUTPUT] , …… , @参数n 数据类型[=默认值] [INPUT | OUTPUT]
AS
    T-SQL程序块
GO
```


8.3 存储过程和函数

```
Create procedure setGrade
    @sno char(10), @cno char(4), @grade numeric(4, 1),
    @st varchar(50) output
AS
    begin transaction
    Declare @cnt int
    select @cnt=0, @st='数据修改'
    Select @cnt=count(*) from SC Where Sno=@sno and Cno=@cno
    if @cnt>0
        update sc set Grade=@grade Where Sno=@sno and Cno=@cno
    else
        begin
            Insert into sc values(@sno,@cno,@grade)
            set @st='数据插入'
        end
    commit transaction
```

```
Create  procedure  setGrade
   @sno  char(10), @cno  char(4),
   @grade numeric(4, 1),@st varchar(50)
output
AS
   begin  transaction
   Declare  @cnt  int
   select @cnt=0, @st='数据修改'
   Select  @cnt=count(*)  from  SC
      Where Sno=@sno  and  Cno=@cno
   if @cnt>0
      update  sc  set Grade=@grade
            Where Sno=@sno and Cno=@cno
   else
      begin
        Insert  into sc values(@sno,@cno,@
grade)
        set @st='数据插入'
      end
   commit  transaction
```

消息
命令已成功完成。

- **过程调用：**

Declare @ret_st varchar(50)

EXEC setGrade '20160001',

'c209', 90, @ret_st output

Print @ret_st

```
-- 调用过程 setGrade
Declare  @ret_st  varchar(50)
EXEC setGrade '20160002',  'c209', 92,
    @ret_st output
Print @ret_st
```

消息
(1 行受影响)
数据修改

```
--  调用过程 setGrade
Declare  @ret_st  varchar(50)
EXEC setGrade '20160003','c209', 100,
    @ret_st output
Print @ret_st
```

消息
(1 行受影响)
数据插入

8.3 存储过程和函数

8.3.2 函数

- **PL/SQL创建函数**

Create or replace function 函数名[(argment [{in|out|in out}] type,...)]
Return return_type
{IS |AS}
<类型、变量说明>
Begin
<函数体，执行部分>
Exception
<可选的异常处理部分>
End;

8.3 存储过程和函数

- **T-SQL创建函数**

create function function_name
([{@parameter_name [as] date_type [=default]}[,...n]])
returns return_data_type
[as]
begin
function_body
return scalar_expression
end

8.3 存储过程和函数

- **T-SQL函数简单示例**

```
create function getGrade( @sno char(10), @cno char(6) )
returns numeric(4,1)
As
begin
   declare @grade numeric(4,1)
   set @grade = -1;
   selecet @grade = grade from sc  where Sno = @sno  and Cno=@cno
   return @grade
End
------------------
Declare  @s1 char(10)='20160001', @c1 char(6)='c209', @g1 numeric(4,1)
Select  @g1 = getGrade(@s1,@c1);
If @g1<>-1
    print '成绩为：'+ convert(char(5), @g1)
Else
    print '为找到课程成绩！'
```

8.3 存储过程和函数

```
-- 创建用户自定义函数 getGrade

create function getGrade( @sno char(10),
@cno char(6) )
returns numeric(4,1)
As
begin
      declare @grade numeric(4,1)
      set @grade = -1;
      select @grade = grade from sc  where
 Sno = @sno  and Cno=@cno
      return @grade
End
```

消息
命令已成功完成。

```
-- 使用自定义函数 getGrade
Declare  @s1 char(10)='20160001', @c1
char(6)='c209', @g1 numeric(4,1)
Select  @g1 = dbo.getGrade(@s1,@c1);
If @g1<>-1
  print  '成绩为：'+ convert(char(5), @g1)
Else
  print  '未找到课程成绩！'
```

消息
成绩为：90.0

```
-- 使用自定义函数 getGrade
Declare  @s1 char(10)='20160004', @c1
char(6)='c209', @g1 numeric(4,1)
Select  @g1 = dbo.getGrade(@s1,@c1);
If @g1<>-1
  print  '成绩为：'+ convert(char(5), @g1)
Else
  print  '未找到课程成绩！'
```

消息
未找到课程成绩！

■ **定义触发器**

PL/SQL:

CREATE TRIGGER <触发器名>
{ BEFORE | AFTER}
<触发事件> ON <表名>
FOR EACH {ROW | STATEMENT}
<触发动作体>;

特殊对象：New, Old

T-SQL:

CREATE TRIGGER <触发器名> ON <表名>
{ FOR | AFTER | INSTEAD OF } <触发事件>
AS
<触发动作体>;

特殊关系：Inserted, Deleted

[例] 定义一个BEFORE行级触发器，为教师表Teacher定义完整性规则“讲师的工资不得低于3000元，如果低于3000元，自动改为3000元”。

```
CREATE TRIGGER Insert_Or_Update_Sal
  BEFORE INSERT OR UPDATE ON Teacher  /*触发事件是插入或更新操作*/
  FOR EACH ROW                        /*行级触发器*/
  AS BEGIN                            /*定义触发动作体，是PL/SQL过程块*/
    IF (new.Job='讲师') AND (new.Sal < 3000) THEN
        new.Sal :=3000;
    END IF;
  END;
```

（T-SQL示例）

8.4 SQL中的事务处理

■ **事务(Transaction)的定义**

- ✓ 构成单一逻辑工作单元的数据库操作序列；
- ✓ 一个不可分割的工作单位；
- ✓ 恢复和并发控制的基本单位。

◆ 一个事务就是将一系列的数据操纵SQL语句作为一个逻辑单元，逻辑单元里面的单个操作要么全做，要么全部不做，以保证数据的完整性。

■ **事务的ACID特性:**

原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）
持续性（Durability ）

8.4 SQL中的事务处理

■ **PL/SQL事务支持的相关语句**

Commit, rollback, savepoint, rollback to savepoint,

✓ PL/SQL未提供事务开始的显式定义语句

✓ 事务的提交与回滚

Commit, rollback

✓ 基于保存点的事务回滚

savepoint sp1;

<SQL的数据更新语句>;

rollback to sp1;

8.4 SQL中的事务处理

■ **T-SQL的事务**

begin transaction [trans_name]，commit transaction [trans_name]，rollback transaction [trans_name] [st_name]，save transaction <st_name>

相关全局变量：@@ERROR, @@TRANSCOUNT

◆ T-SQL事务的分类:

✓ 显式事务：用begin transaction定义

✓ 隐式事务：通过set implcit_transaction on打开隐式事务模式，下一个语句自动开启一个新事务，事务结束后的下一个语句又将开启一个新事务。

✓ 自动提交事务（默认模式）：将每条SQL语句视为一个事务，执行成功则自动提交；失败则自动回滚。

第8章 数据库编程

- **本章思考题：**
  嵌入式SQL是数据库应用系统开发的主要手段，主语言的数据处理能力非常强大，为何还需要使用过程化SQL?

- **本章作业：**
  教材 习题2（要求用T-SQL完成!）